Universität Bremen
FB 3 – Informatik
Prof. Dr. Rainer Koschke
TutorIn: Sabrina Wilske

Software–Projekt 2 2013

VAK 03-BA-901.02

Testplan

IT_R3V0LUT10N

Sebastian Bredehöft	sbrede@tzi.de	2751589
Patrick Damrow	damsen@tzi.de	2056170
Tobias Dellert	tode@tzi.de	2936941
Tim Ellhoff	tellhoff@tzi.de	2520913
Daniel Pupat	dpupat@tzi.de	2703053

Abgabe: 22. Dezemebr 2013 — Version 1.0

Inhaltsverzeichnis

Version und Änderungsgeschichte

Version	Datum	Änderungen
1.0	18.12.2013	Dokumentvorlage als initiale Fassung kopiert
1.1	21.12.2013	Bis auf Testfälle und Testzeitplan vervollständigt

1 Einführung (Sebastian)

1.1 Zweck

Der Testplan bietet einen Überblick über die geplanten Tests und dient u.a. als Anleitung für die Tester. Die Software soll dabei ausführlich auf Funktionalität getestet werden.

Im Testplan wird festgelegt, wie man welche Komponenten testet. Dazu wird außerdem definiert, welchen Umfang die Tests haben sollen und wann ein Test erfolgreich ist und wann nicht.

Während der Implementierungsphase werden wir uns nach dem Testplan richten und ihn gegebenenfalls weiterführen und vervollständigen.

1.2 Umfang

Der Testplan entspricht der vereinfachten Form des *IEEE Standard for Software Test Documentation 829-1998*.

1.3 Beziehungen zu anderen Dokumenten

Dieser Testplan bezieht sich zum einen auf die Anforderungsspezifikation, da dort die Systemeigenschaften und Systemattribute spezifiziert wurden. Die Testfälle werden auf Grundlage der dortigen Anwendungsfälle entwickelt.

Außerdem gibt es Referenzen zur Architekturbeschreibung, da in dieser die Module und Komponenten definiert wurden, die in diesem Dokument getestet werden sollen.

1.4 Aufbau der Testbezeichner

Der Aufbau der Testbezeichner richtet sich nach folgendem Schema:

- Die ersten beiden Buchstaben geben die Art des Tests vor. Dabei unterscheiden wir zwischen vier verschiedenen Testarten:
  - Komponententests = KT
  - Integrationstests = IT

  - Funktionstests = FT
  - Leistungstests = LT
- Die Nummer steht für die jeweilige Testfallnummer
- Optional: in alphabetischer Reihenfolge werden hier Variationen oder untergeordnete Testfälle definiert

Nach diesem Schema sieht ein Testbezeichner nun folgendermaßen aus:

IT-3-A: Integrationstest, Nr. 3, Variante 1

1.5 Dokumentation der Testergebnisse

Zu jedem Testfall wird ein kurzes Testprotokoll angefertigt. Dieses beinhaltet den Ablauf des Testfalls und die möglichen Komplikationen, die bei der Durchführung entstehen können. Dann werden die Resultate des Testfalls bestimmt und eventuell gefundene Fehler beschrieben.

1.6 Definitionen und Akronyme

1.7 Referenzen

IEEE Standard for Software Test Documentation 829-1998
http://standards.ieee.org/findstds/standard/829-1998.html

2 Systemüberblick (Sebastian)

Das System besteht aus der Server- und der Clientkomponente. Die konzeptionelle Sicht der Architekturbeschreibung (vgl. Abschnitt 3 der Architekturbeschreibung) dient als Grundlage für den Testplan, da dort die verschiedenen Komponenten beschrieben werden.

Auf der Serverseite gibt es die Komponenten `Communication, BusinessLogic` und `Persistence` (vgl. Abbildung 3: Konzeptionelle Sicht Server; Architekturbeschreibung).

Die Clientseite besteht aus den Komponenten `Communication, Model` und `User Interface` (vgl. Abbildung 4: Konzeptionelle Sicht Client; Architekturbeschreibung).

Da starke Abhängigkeiten zwischen all diesen Komponenten bestehen, ist es wichtig, dass diese Komponenten fehlerfrei funktionieren.

Es sollten die Hofmeister-Konzepte verwendet werden. Benutzt am besten Diagramme aus der Architekturbeschreibung.

Dieser Abschnitt ist nicht einfach nur Copy&Paste der Architekturbeschreibung. Es gilt

einerseits, Redundanzen soweit wie möglich zu vermeiden, und andererseits, dieses Dokument so selbsterklärend wie möglich zu machen.

2.1 Module der Anwendungsschicht und deren Funktionen

Muss in SWP-2 ausgefüllt werden

Hier solltet ihr dann die einzelnen Module aus Sicht des Tests weiter verfeinern. Auch wäre noch eine Grafik angebracht, die die Abhängigkeiten der Subsysteme/Module zeigt, sowie eine kurze Beschreibung dazu.

2.1.1 Server

2.1.2 Client

3 Merkmale (Daniel)

Zu testende Merkmale sind in erster Linie Funktionen, die in den Mindestanforderungen enthalten sind. Dabei ist zu beachten, dass die Funktionen des Lesers für die Website und der App getestet werden müssen, während sich die anderen Testmerkmale auf die Website beziehen. Beide sind im Folgenden aufgelistet:

1. Benutzerrechte

   1.1 Administrator
   1.2 Bibliothekar
   1.3 Leser
   1.4 Gast

2. Administrator

   2.1 Bibliothekar hinzufügen
   2.2 Bibliothekar löschen
   2.3 Bibliothekar ändern

3. Bibliothekar

   3.1 Medium hinzufügen
   3.2 Medium löschen
   3.3 Medium ändern
   3.4 Medium ausleihen
   3.5 Medium zurücknehmen
   3.6 Abgabedaten und Mahngebühren ändern
   3.7 Vormerkungen anzeigen
   3.8 Übersicht über verliehene Bücher(Versäumnisse, Mahngebühren)
   3.9 Statistiken anzeigen

4.Leser

   4.1 Medium suchen
   4.2 Medium anzeigen

4.3 Medium vormerken

Dazu kommt noch der Gast, der unangemeldet nur suchen kann und man sollte die Unterklassen von Medium noch einzeln testen, ob diese die geforderten Attribute und Funktionen enthalten.

3.0.3 Funktionale Anforderungen

Besonders wichtig sind die Funktionen des Bibliothekars (3.1-3.5). Die sollten gut getestet werden, da es beim Ausleihvorgang nicht zu Problemen kommen soll, sodass irgendwo Bücher verschwinden oder Ausleiher oder Bücher verwechselt werden. Auch wichtig ist die Benutzerunterscheidung, damit Leser nicht irgendwas löschen oder hinzufügen. Außerdem sollte man die Suche und das Anzeigen der Bücher ausgiebig testen, da dies die Hauptfunktionen des Lesers sind.

3.1 Nicht zu testende Merkmale

Nicht zu testende Merkmale sind in erster Linie alle trivialen Funktionen. Zudem brauchen bereits implementierte Funktionen, wie `Leser verwalten, Backup` und `Restore, Buchaufkleber drucken, Leserausweise drucken, Import und Export von Buch- und Leserdaten` nicht mehr getestet werden, sofern man diese nicht verändert. Da wir `Buch verwalten` noch verändern, da wir mit mehreren Medientypen arbeiten, ist dieses wie in **??** beschrieben noch zu testen.

4 Abnahme- und Testendekriterien(Daniel & Sebastian)

Fehler werden in eine Kategorie eingeordnet und erhalten entsprechende Fehlerwerte. Aus diesen Fehlerwerten ergeben sich Prioritäten, die die Reihenfolge der Fehlerbehandlung angibt. Das Testen wird beendet, wenn der berechnete Fehlerwert aller Fehler pro 1000 Zeilen Code unter dem Wert 10 liegt und die Software nicht beeinträchtigt wird, d.h. es keinen Fehler der Fehlerklasse `Mittel` oder höher gibt.

Testabdeckung Die Testabdeckung soll so hoch wie möglich sein. Für ein stabiles System spricht, das die Testabdeckung in systemkritischen Bereichen soweit vollständig ist. Jeder Fehler in diesem Bereich kann das System zum Absturz bringen und muss somit verhindert werden. In anderen Bereichen die das laufende System bei einem Fehler weniger beinträchtigen wird die Testabdeckung nicht so vollständig sein, wie in kritischen Bereichen.

Fehlerbewertung:
Die nachfolgende Tabelle spezifiziert die Auswirkung eines Fehlers, durch die man diese nach Priorität einordnen kann.

Fehlerkl.[1]	Beschreibung	Wert
Leicht	Unwesentliche Fehler, die den Programmablauf nicht beeinträchtigen, aber trotzdem behandelt werden sollten.	1
Mittel	Fehler in dieser Art haben Auswirkungen auf den Programmablauf. Dieser beeinträchtigt aber nicht die grundlegenden Funktionen.	10
Schwer	Fehler der Klasse ,,Schwer" beeinträchtigen die Funktionsfähigkeit des Systems sehr stark und müssen sofort behandelt werden.	20
Fatal	Diese Fehler machen den Programmablauf unmöglich und können zum Absturz des Systems führen.	100

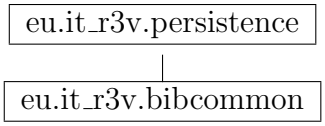
5 Vorgehensweise (Sebastian)

5.1 Komponenten- und Integrationstest

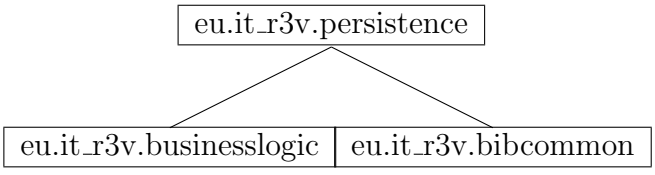
Die beiden Komponenten Server und Client werden bezüglich der Integrationstests zunächst unabhängig voneinander getestet und erst bei wenn das sichere Laufen der einzelnen Komponente gewährleistet ist, wird das System als ganzes getestet.

Server

Es wird zuerst die Persistenz mit `eu.it_r3v.bibcommon` und `eu.it_r3v.persistence` getestet:

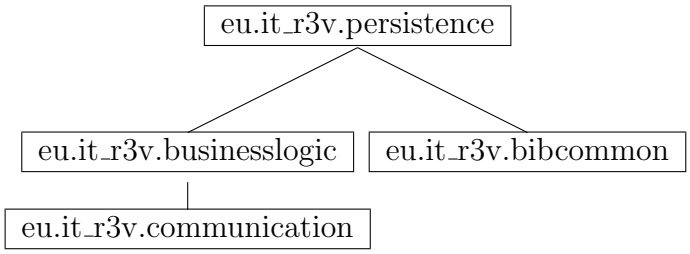


Nun folgt das Zusammenspiel mit der BusinessLogic `eu.it_r3v.businesslogic`:



Darauf folgt nun die Kommunikation `eu.it_r3v.communication` mit den vorigen Komponenten:

[1]=Fehlerklasse

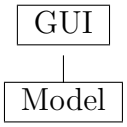


Somit ist der Server als ganzes zu testen, da jede einzelnen Komponenten mit ihren jeweiligen Abhängigkeiten getestet wurden.

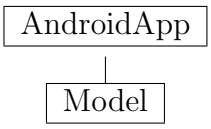
Client

Hier sind nun zwei verschiedene Komponenten für die Darstellung auf dem jeweiligen Endgerät vorhanden:

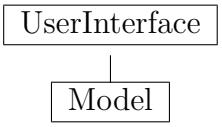
Einmal `GUI` welche zusammen mit dem Model getestet wird und sich an die Browserdarstellung richtet:



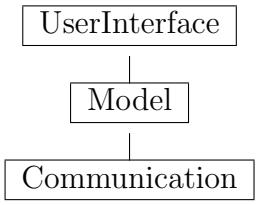
Und einmal `Android App` mit dem Model, welche sich an mobile Geräte richtet:



Diese beiden Tests werden zusammengefasst zu `User Interface` und zusammen mit der nächsten Ebene, dem Model getestet:



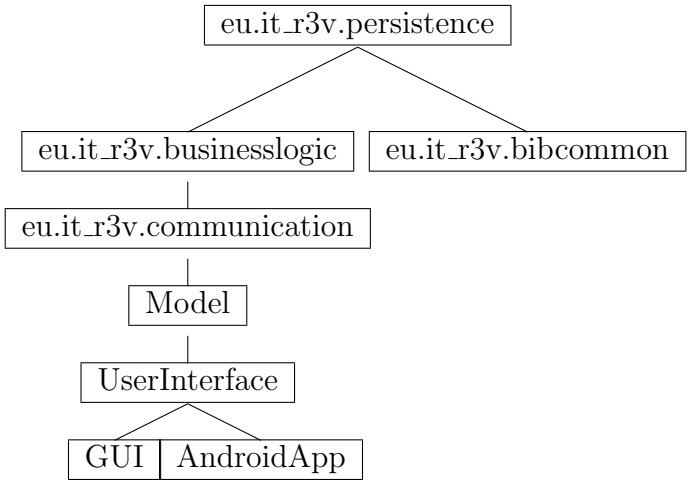
Darauf folgt nun der Test mit der Komponente `Communication`:



Nun ist das Zusammenspiel der Komponenten des Clients gewährleistet.

Server + Client

Um die kompletten Systemkomponenten zu testen werden jetzt der komplette Client und der komplette Server zusammen getestet:



5.2 Funktionstest

Die Funktionstest sind durch jene Anwendungsfälle aus der Anforderungsspezifikation vorgegeben. Jede dieser Funktionen muss durch Tests gedeckt sein.

6 Aufhebung und Wiederaufnahme (Daniel)

Wir werden Tests unterbrechen, wenn ein gewisser Wert überschritten wird, welcher über die Tabelle **??**berechnet wird. In diesem Fall werden wir sofort wieder mit der Implementierung anfangen. Da wir mit der Bottom-up Strategie testen, werden wir bei Fehlern in der unteren Schicht einen niedrigeren Wert nehmen.
Bei Fehlern der Data setzen wir einen Wert von 10, bei Fehlern in der Logik einen Wert von 20 und bei Fehlern, welche die GUI betreffen, einen von 40 und bei den restlichen Faktoren einen von 100.

Sollten die Fehler behoben sein, testen wir noch einmal alle Komponenten, die mit den veränderten interagieren.

7 Hardware- und Softwareanforderungen (Daniel)

7.1 Hardware

Als Hardware stehen uns unsere Notebooks und Smartphones, sowie die Unirechner zur Verfügung. Dabei haben wir alle geforderten Betriebssysteme mindestens einmal auf unseren Notebooks installiert, sodass wir auf jeden Gerät testen können. Wir besitzen ebenfalls einen PC, der über Windows 2000 läuft, darüber testen wir auch noch, da dies den Rechnern des Kunden entspricht. Da nur Android Unterstützung gefordert ist, werden wir die App über unseren vorhandenen Smartphones, die Android haben, testen.

7.2 Software

Als Software benutzen wir in der Eclipse Umgebung JUnit-Tests. Diese werden in Form von BlackBox- und WhiteBox-Tests implementiert. Die App werden wir mithilfe eines Android Emulators testen.

8 Testfälle

Dies ist der wichtigste (und vermutlich umfangreichste) Teil des Testplans. Hier wird genau aufgelistet, **was wie** *und* **von wem** *getestet wird. Das spätere Testen besteht dann einfach aus einer Durchführung dieser Tests.*

Zu jedem Testfall muss es eine **Testfallspezifikation** *geben (außer Komponententests, siehe unten). Schwerpunkt sollen hier Integrations- und Leistungstests sein. Welche Arten von Integrationstests und Leistungstests seht Ihr vor und wie wollt Ihr diese genau ausgestalten?*

8.1 Komponententest

Wir haben hier alle Klassen aufgelistet, welche wir testen wollen. Dabei werden wir keine abstrakten Klassen und Interfaces testen. Vorgegebene Klassen, welche bereits funktionieren, werden wir nicht testen. Exceptions testen wir nicht einzeln, sondern diese werden mit den zugehörigen Methoden getestet.

Klasse	Implementierer	Tester	Testart
AsyncBookTask	Patrick	Tim	Blackbox
BookAdapter	Patrick	Daniel	Blackbox
MainActivity	Patrick	Sebastian	Blackbox
Network	Patrick	Tobias	Blackbox
ShowBookActivity	Patrick	Tim	Blackbox
Admin	Tim	Daniel	Blackbox
Medium	Tim	Patrick	Blackbox
Reader	Tim	Tobias	Blackbox
Librarian	Tim	Daniel	Blackbox
Config	Tim	Sebastian	Blackbox
AdministrationHandler	Daniel	Patrick	Blackbox
BorrowHandler	Daniel	Tim	Blackbox
MediumHandler	Daniel	Tobias	Blackbox
LibrarianHandler	Daniel	Tim	Blackbox
ReaderHandler	Daniel	Tim	Blackbox
Data	Daniel	Tobias	Blackbox
AddLibrarianForm	Tobias	Patrick	Blackbox
AddMediumForm	Tobias	Tim	Blackbox
AddReaderForm	Tobias	Sebastian	Blackbox
Administration	Tobias	Patrick	Blackbox
AuthBackingBean	Tobias	Tim	Blackbox
MediumListDataModel	Sebastian	Patrick	Blackbox
MediumTable	Sebastian	Daniel	Blackbox
ReaderTable	Sebastian	Tobias	Blackbox
ChangeMediumForm	Sebastian	Tim	Blackbox
ChangeReaderForm	Sebastian	Daniel	Blackbox
TableDataModel	Sebastian	Patrick	Blackbox
BibServices	Sebastian	Tim	Blackbox

Tabelle 1: Komponententests

8.2 Integrationstest

Die Integrationstests werden hier genauer beschrieben: Welche Klassen sind beteiligt? Wie ist der Zustand des Systems vor Beginn? Welche Eingaben werden getätigt? Welche Ergebnisse/welches Verhalten wird erwartet?

Aufbau einer Testfallspezifikation:

1. *eindeutiger Testfallbezeichner (entspricht Namenskonvention aus Abschnitt 1.4)*
2. *Testobjekte: welche Komponenten werden getestet?*
3. *Eingabespezifikationen (Eingaben des Testfalls)*
4. *Ausgabespezifikationen (erwartete Ausgaben)*
5. *Umgebungserfordernisse (notwendige Software- und Hardwareplattform sowie Testtreiber und -rümpfe)*
6. *besondere prozedurale Anforderungen (Einschränkungen wie Zeitvorgaben, Belastung oder Eingreifen durch den Operator)*
7. *Abhängigkeiten zwischen Testfällen*

8.3 Funktionstest

Muss in SWP-2 ausgefüllt werden

Die Funktionstests basieren auf den Anwendungsfällen. Diese müssen hier nun konkretisiert werden, um einen Testfall zu erhalten. Es müssen also konkrete Werte für Ein- und Ausgabe festgelegt werden. Auch das Verhalten im Fehlerfall sollte getestet werden.

Die Anwendungsfälle müssen nicht alle wiederholt werden. Sie werden schließlich in der Anforderungsspezifikation spezifiziert. Hier genügt eine tabellarische Auflistung aller zu testenden Anwendungsfälle, deren konkrete Ein- und Ausgaben sowie die Umsetzung (automatisiert oder manuell und falls manuell, durch welche Art von Benutzer?). Insbesondere müsst Ihr an dieser Stelle klären, welche Varianten von Anwendungsfällen getestet werden. Falls Varianten oder gar ganze Anwendungsfälle nicht getestet werden sollen, dann wird hier eine plausible Begründung erwartet.

8.4 Leistungstest

Muss in SWP-2 ausgefüllt werden

Die Leistungstests prüfen die Leistungsanforderungen, wie z.B. Reaktionszeiten, das Verhalten unter extremen Bedingungen, bei großen Datenmengen etc. . .

8.4.1 Härtetest

Muss in SWP-2 ausgefüllt werden

8.4.2 Volumentest

Muss in SWP-2 ausgefüllt werden

8.4.3 Sicherheitstest

Muss in SWP-2 ausgefüllt werden

8.4.4 Erholungstest

Muss in SWP-2 ausgefüllt werden

Hier wird geprüft, ob sich das System von Fehlerzuständen auch wieder erholt. Es werden also gezielt Fehler provoziert, um die Korrektheit der Systemreaktion herauszufinden.

9 Testzeitplan

Muss in SWP-2 ausgefüllt werden